View metadata, citation and similar papers at core.ac.uk
brought to you by CORE

Plan 9 web サーバ Pegasus

経営学部教授　有　澤　健　治

序文

Pegasus とは Plan 9 の標準 web サーバを基に筆者が開発した新しいタイプの web サーバである。何が新しいのかはこの論文の中で解説する。このサーバは2002年の元旦よりインターネットの世界に船出し今日に至っている。Pegasus の名前はこの年の干支に因んで付けられた。

Plan 9 と言うのはベル研究所が1995年に公開した新しいタイプのオペレーティングシステムである。ベル研究所は様々な分野において輝かしい業績を残しているが、コンピュータ科学の分野では UNIX を生み出した事で有名である。UNIX を開発した同じグループが、ネットワーク時代に対応した新種のオペレーティングシステムとして Plan 9 を送り出したのである。

オペレーティングシステムがネットワーク環境と旨く適合するにはプロセス（実行中のプログラム）が見る名前空間（ファイルの名前のなす空間）が全てのプロセスで同じであってはいけない。プロセス毎に編成される必要がある、と彼らは考えた。Plan 9 はプロセス毎に私的な名前空間を割り付けることができるユニークなオペレーティングシステムである。

web サーバは現代における第一級のサーバアプリケーションである。UNIX であれ、Windows であれ、そこで動く web サーバは様々なセキュリティの問題を抱えている。重要な問題の幾つかは、サーバが見る名前空間に由来している。Plan 9 で動く web サーバは、それらの問題を解決できる潜在的な可能性を秘めているのである。

ベル研究所が Plan 9 に添えた web サーバは平凡なものであった。彼らはまるで web サーバには興味が無いかのように見える。実際彼らはもっともっと面白い発明に熱中しているのである。しかし、web サーバには新奇性はないにせよ、第一級のサーバアプリケーションである。そして病んでいるのである。問題は解決する必要がある。Pegasus はそのために開発された。

以下に紹介する記事は、筆者の web サーバの記事（注１）に多少の修正を加えたものである。Pegasus の最新版は2.0であり、注２の URI から取り寄せられる。興味のある読者は

Plan 9 をインストールした上で使用されたい。また、この１年間の Pegasus の発展に興味があれば注３の URI を参照願いたい。

注１：http://plan9.aichi-u.ac.jp/pegasus/man-2.0/concept.html

注２：http://plan9.aichi-u.ac.jp/netlib/pegasus-2.0.tgz

注３：http://plan9.aichi-u.ac.jp/pegasus/

１．ドキュメントの管理主体に基づく名前空間の分離

通常の web のサーバ

通常の web のサーバにおいてはサーバルートと言う概念は本質的な意味を持っていません。例えば Apache では httpd.conf の中で ServerRoot を指定しますが、これは単にアクセス記録などを置く場所を意味しているに過ぎません。

通常の web サーバではサーバルートはアクセス制限に対して何の意味も持っていません。この事は CGI を使用した場合に直ちに問題になります。即ち web のサーバは一見するとクライアントからのアクセス可能なファイルを制限しているように見えても、CGI プログラムからはシステムの全てのファイルが見えてしまいます。この事は重大なセキュリティ上の問題をもたらしかねません。そのためにエンドユーザによる CGI の使用は禁止されるかあるいは厳重な統制下に置かれるのが普通です。

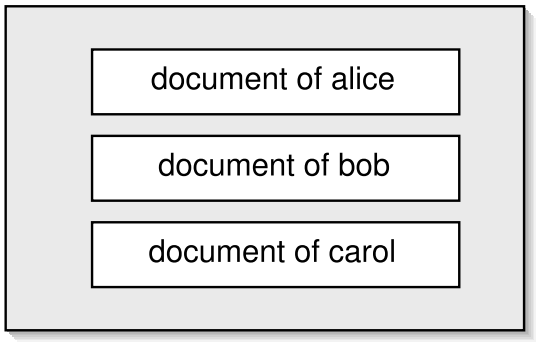

図１：伝統的な web サーバの名前空間
real space=service space

図１に通常の web サーバにおける名前空間の様子を概念的に示します。

外枠の矩形領域が実空間（real space）を表しています。そして外枠と一致するうすい灰色で示した矩形領域がサービス空間（service space）です。実空間とはコンソールから見えるファイルの集合であり、（Plan 9 以外の OS では）システムのファイルの全体を表してい

ます。サービス空間とは web のサーバがサービスを行っているファイルの集合です。これはまた CGI プログラムから見えるファイルの集合でもあります。通常の web のサーバでは実空間とサービス空間が正確に一致しています。

ドキュメント空間（document space）は web ページを構成するファイルの集合です。alice のドキュメント空間はブラウザからは /~alice でアクセスされるファイルの集合です。サーバ上には様々な人々が作成するドキュメントがありますが、これらはサービス空間の中にあるので CGI からは隠されていません。

Plan 9 の標準 web サーバによる進展

サーバルートの概念に本質的な意味を最初に与えたのは Plan 9 第 2 版の標準 web サーバです。Plan 9 がプロセス毎に名前空間を編成できる事を利用して、このサーバでは web サーバのサービス空間を（従って CGI プログラムから見える空間も）、サーバルートの下に綴じ込めることができました（図 2 ）。

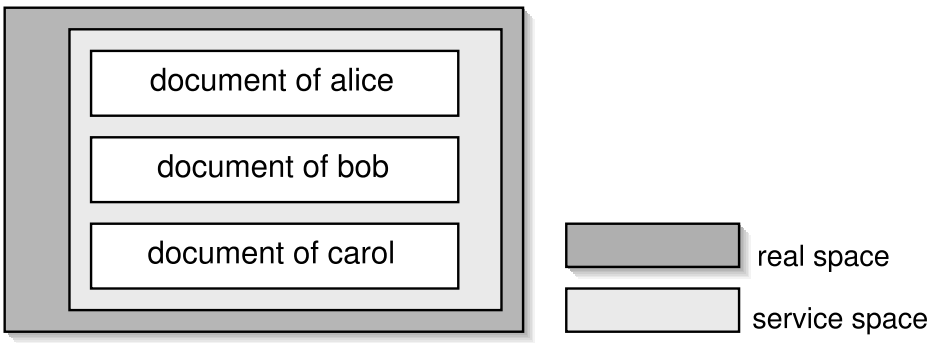

図 2 :Plan 9 の標準 web サーバで実現された名前空間
real space>service space

従ってこのサーバの下では CGI プログラムから見えるファイルは、指定されたサーバルートの下にあるファイルだけです。つまり CGI プログラムはサーバルートによって指定された名前空間の中に完全に綴じ込められるのです。図 2 はこの様子を示しています。濃い灰色で示された部分がサービス空間の外にあるファイルの集まりです。この部分は CGI プログラムから本質的に隠されています。

Pegasus によって実現された web サーバの名前空間

Pegasus は、これまでの Plan 9 上の web サーバのこの素晴らしいアイデアを引き継ぎ、さらに発展させました。

一般に１つのサーバ（some.dom.com）には様々な人々が作成する web のドキュメントが存在し、それらはサーバにおけるそれらの人々の位置付けに従って次のようにいろいろな方法でクライアントからアクセスされます。

```
http://some.dom.com/pathname          # 実ホストのドキュメント
http://some.dom.com/~alice/pathname   # ユーザドキュメント
http://other.dome.com/pathname        # 仮想ホスト（IP が異なる）のドキュメント
http://virtual.dom.com/pathname       # 仮想ホスト（IP が同じ）のドキュメント
```

ここに pathname はドキュメントルートからみたドキュメントへのパスです。

これまでの web サーバの大きな問題点は、サービス空間がこれらの間で共有される事にあります（図1，図２）。この事は、ある人が作成した CGI プログラムを通じて、他の人のファイルにアクセスできる可能性がある事を意味しています。従ってドキュメントを作成し公開する人々の間の相互干渉の可能性をもたらします。

注釈：コンピュータの仕組みに例えると、全てのプロセスがアドレス空間を共有している初期のパソコンとよく似ている。プロセスごとの論理アドレスに相当する機能を提供していないのである。UNIX が PC に移植できたのは i386が MMU との組み合わせで論理アドレスをプロセスに提供できるようになったからである。

この問題は図３ a、図３ b に示すように、サービスを行う web のサーバが、ドキュメントごとの名前空間の中でサービスを行うことによって解決されます。そしてこれは Pegasus によって初めて解決されました。

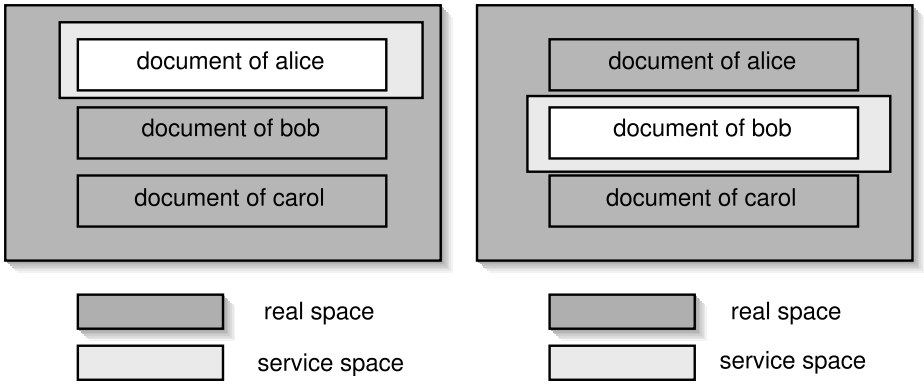

図３ a　　図３ b

ドキュメントの管理者ごとに独立した名前空間が与えられる必要がある
図３ a は alice のドキュメントをサービスしているときの Pegasus のサービス空間
図３ b は bob のドキュメントをサービスしているときの Pegasus のサービス空間である

2. Pegasus における名前空間の再編成

話をもう少し具体的にし、次のように３人の役割を仮定しましょう。

alice を実ホストのユーザ

bob を実ホストの web ドキュメントの管理者

carol を仮想ホスト（名前を car としましょう）のドキュメントの管理者とします。

この場合には /sys/lib/http.rewrite で

```
http://car          */usr/carol/www
/                   */usr/bob/www
```

と設定します。alice のような一般ユーザは設定なしにホームページが持てます。すなわち、

```
/~alice             */usr/alice/web
```

が暗黙のうちに仮定されます。

そしてシステムの管理者（たぶん bob でしょう）は /lib/namespace.httpd で web サーバのサービス空間を定義します。するとクライアントからアクセスがあるたびに、このサービス空間の中に、要求があったドキュメントの管理者のファイルだけが、マージされます。その結果クライアントから alice のドキュメントに要求があった場合には図４に示されるサービス空間とドキュメント空間が実現されます。（但し、この図では /lib/namespace.httpd による編成部分は省略されています。この部分は薄い灰色の領域に含まれます。）

それ以外は見えないわけですから、お店の経営に例えると、alice は bob と carol との寄り合い所帯のお店ではなく、独立したお店を与えられた事になります。この事が、CGI におけるトラブルを避ける基礎になります。

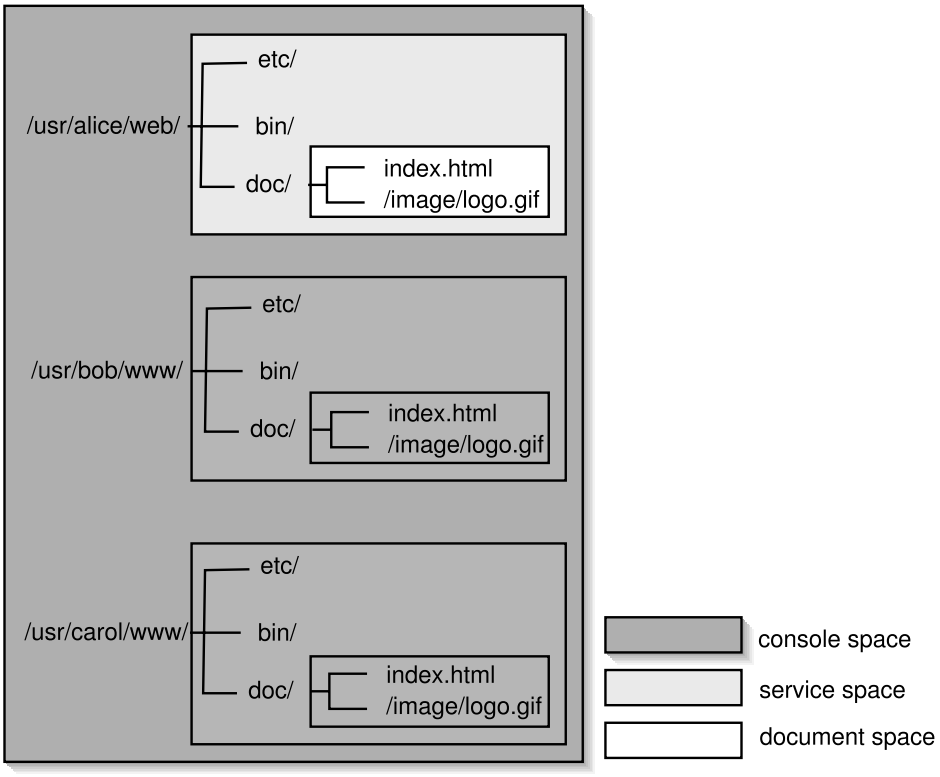

図4．Pegasus で実現された web サーバの名前空間
ドキュメントの管理者ごとに独立した名前空間が与えられる

3．ファイルのアクセス保護

web サーバ上のファイルは、他のシステムユーザの ftp や telnet などを通じたアクセス、および他のユーザが作成した CGI を通じた不正アクセスから保護される必要があります。この問題は web サーバにとって頭の痛い問題の一つでした。

例えば alice が CGI で使用するあるファイル data は alice 自身と alice の作成した CGI だけからの読みとりを許したい場合があります。

UNIX では web サーバは nobody としてサービスを行っているので、alice は nobody に対して data の読みとりを許可しなくてはなりません。すると他のユーザの CGI からも data が読みとられることになります。

Windows では web サーバは LocalSys としてサービスを行っています。LocalSys は UNIX の root と同等の特権を持っています。この場合にはどのようになるのでしょうか…
Pegasus ではこの問題を次の様に解決します。

1．サーバをユーザ web として実行する。

ただし web は実際のユーザではありません。(従ってファイルを所有する必要はありません。)

2．/adm/users に alice のグループメンバとして web を追加する。即ち、

alice : alice : web

3．alice は data のアクセスモードを

--rw-r-----alice web .... data

に設定する。

注意：アクセス保護の問題がこんな簡単な方法で解決できるのは Pegasus のサービス空間がユーザ毎に綴じ込められているからです。

UNIX における救いは、この問題の解決の必要性を認識し、解決しようと努力している人々が存在することです。2つのアプローチを紹介します。

CGI wrapper

UNIX では現在の所、この問題を CGI wrapper によって解決しようとしています（例えば http://download.sourceforge.net/cgiwrap)。すなわち web サーバは CGI wrapper を通じてのみ CGI にアクセスし、CGI wrapper には root による SUID ビットが立てられます。CGI は指定された場所に置くことを求められます。CGI wrapper が CGI にアクセスする時には root 特権を利用して CGI ファイルのオーナーに変身してアクセスします。CGI ファイルに直接 SUID ビットを立てないのは、あまりにも危険だからです。(筆者は SUID は UNIX の失敗作であると思っていましたが、こういう所で役に立つとは考えてもみませんでした。)

UNIX の CGI wrapper と Pegasus とを比較すると Pegasus の方が安全性が高いと言えます。CGI wrapper では下手な CGI を作ると、そのユーザの全てのファイルを危険にさらします。Pegasus ではユーザ web に対して書き込みアクセスを許しているファイルだけが危険にさらされます。(Windows ではシステムの全てを危険にさらす !)

管理面でも Pegasus の方がはるかに簡単です。管理者側から一切の管理が必要ありません。単にユーザ web として httpd を実行すればよいだけです。そして CGI はどこにおいても構いません。

9.2.u と v9fs

UNIX でも Plan 9 の「プロセスごとに編成できる名前空間」を実現しようとするプロジェクトがあります。

第一段階として open などの標準入出力ライブラリを書き直し、新しいライブラリを使用したプログラムは Plan 9 的な振る舞いをするようにします。しかしこのままでは古いライ

ブラリを使用したプログラムからの保護が効かないので、やがては変更はカーネルレベルにまで持っていく計画です。(完全な置き換えには10年ほどかかるでしょう。そしてその頃になったらUNIXにもPegasusと同じコンセプトに立ったwebのサーバが実現するでしょう。)

カーネルレベルで名前空間を閉じこめない限り、ユーザレベルのCGIの完全な自由はありません。しかし、CGIをスクリプトに限定し、新しいライブラリを使用したスクリプト言語の使用を許すだけでも非常に大きな前進です。詳しくは次のURIを参照してください。

http://www.ddj.com/documents/s=1782/ddj0112a/0112a.htm

さらに詳しい論文がPS形式で出されています。(筆者のサーバに置いてあります:

http://plan9.aichi-u.ac.jp/pegasus/man-2.0/ref/minnich-dobbs01.ps)

最近、Plan 9のファイルシステムとのインターフェースの仕組みが発表されました。

(http://sourceforge.net/projects/v9fs/)

4. 自由度の高い仮想ドキュメント環境

Pegasusの現在の目標は高度な安全性と高度な自由度を備えた仮想ドキュメント環境をユーザに提供することにあります。

Pegasusは仮想ドキュメント環境を実行ハンドラによって実現します。CGIは実行ハンドラの特殊な設定にすぎません。実行ハンドラの内容の設定は完全にドキュメントの管理者に任せられているので、彼らは自分たちのニーズに応じた内容を盛り込むことができます。

実行ハンドラとはリクエストされたファイルのパスパターンに応じてその処理を行うプログラムの事です。実行ハンドラはApacheなどでは、httpd.confの中で可能な設定の一覧がリストアップされ、その中からコメントを外すことによってしか選択できません。Apache/1.3.27の場合については以下の5つです。

```
#AddHandler cgi-script.cgi
#AddHandler server-parsed.shtml
#AddHandler send-as-is asis
#AddHandler imap-file map
#AddHandler type-map var
```

Pegasusではドキュメントの管理者はサービス空間の中の/etc/handlerの中で、クライアントから要求されたファイルのパスパターンとそれを処理するプログラムの関係を定義します。

プログラムはサーバーコードとは独立しており、多くの場合、簡単なスクリプトなので、短時間に開発できます。以下に筆者のサーバ(http://plan9.aichi-u.ac.jp)の実行ハンドラの

設定内容を例示します。

#path	mimetype	unused	execpath arg...
/netlib/*/index.html	text/html	0	/bin/ftp2html
*.http	-	0	$target
*.html	text/html	1	$target
*.dx_html	text/html	0	/bin/dx $target

リクエストされたファイルを処理するプログラムが、リクエストされたファイル自体であれば所謂 CGI プログラムになります。

特殊な拡張子のファイルあるいは特殊な場所にあるファイルに対して特殊なプログラムで処理する事もできます。Server Side Include の機能も実行ハンドラを使用して実現できます。また、FTP サーバのようなサービスを web サーバで代行する場合にはファイルの一覧をクライアントに示す必要があるでしょう。このような場合には FTP 用のファイルが置かれているディレクトリに対して自動的にファイルの一覧を表示するプログラムを実行させるようにします。

Pegasus の実行ハンドラの考え方は広い応用を持つ事が想像できますが、ドキュメント管理者にこれだけの自由が与えられるのは、ここでの設定が、他人のドキュメントに影響を与えないからです。